Combined Array and ADC Structural Test for RRAM-based Multiply-and-Accumulate Circuits

Serlis, E. A.; Xun, H.; Arapidis, E.; Gebregiorgis, A.; Taouil, M.; Hamdioui, S.; Fieback, M.

DOI
10.1109/ITC58126.2025.00076

Publication date
2025

Document Version
Final published version

Published in
Proceedings of the 2025 IEEE International Test Conference (ITC)

Combined Array and ADC Structural Test for RRAM-based Multiply-and-Accumulate Circuits

Emmanouil Anastasios Serlis[1], Hanzhi Xun[1], Emmanouil Arapidis[1], Anteneh Gebregiorgis[1]
Mottaqiallah Taouil[1,2], Said Hamdioui[1,2], Moritz Fieback[1]
[1]Computer Engineering Laboratory, Delft University of Technology, Mekelweg 4, 2628CD, Delft, The Netherlands
[2]CognitiveIC, Van der Burghweg 1, 2628CS, Delft, The Netherlands
Email: {e.a.serlis, h.xun, e.arapidis, a.b.gebregiorgis, m.taouil, s.hamdioui, m.c.r.fieback}@tudelft.nl

2025 IEEE International Test Conference (ITC) | 979-8-3315-7041-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/ITC58126.2025.00076

***Abstract*—Compute-in-memory (CIM) AI accelerators using non-volatile memories like RRAM enable energy-efficient edge inference by executing Multiply-Accumulate (MAC) operations directly in memory in a single cycle. These designs modify memory cells and analog-to-digital converters (ADCs), introducing faults not seen in standard memories. We present the first structural testing methodology and framework for RRAM-based CIM MAC circuits, including defect and fault models for memory cells and ADCs. Our robust inference-driven tests exercise full MAC functionality, significantly reducing test time compared to traditional methods, and integrating cell and peripheral testing to ensure high reliability, defect coverage, and operational efficiency.**

***Index Terms*—Structural test, Multiply-and-accumulate, Computation-in-memory, RRAMs**

I. Introduction

CIM AI accelerators address the Von Neumann bottleneck by improving performance and energy efficiency via analog computing in large macros [1]. They combine circuit components like ADCs and memristive RRAM devices, with the latter exhibiting unique manufacturing defects [2, 3]. CIM systems also experience novel fault mechanisms that differ from those in traditional memories [4], demanding dedicated structural testing for both RRAM and data converters. However, current state-of-the-art testing solutions are mostly functional [5, 6], rely on abstract fault models, and typically do not include structural testing of data converters within CIM macros.

The proposed testing methodology bridges the aforementioned gap, with multi-level column patterns that exploit the inherent parallel computation architecture of RRAM CIM circuits. The main contributions of the paper are to:

- Introduce a structural test development method for RRAM-based MAC CIM.
- Apply the method to derive the fault space for array and ADC faults.
- Demonstrate that the method results in tests that have a high fault coverage and low test-time overhead.

The remainder of the paper is structured as follows. Section II provides the required background. Section III describes the proposed methodology. Sections IV and V describe defect and fault models for RRAMs and ADCs. Section VI develops test patterns for each defective part. Section VII offers a comparison of the proposed methodology with the state-of-the-art.

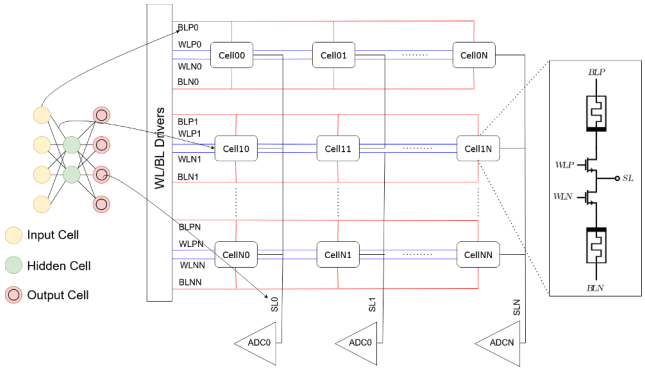

Fig. 1: Mapping of Neural Network to Crossbar of N rows and M columns

II. Background

A. RRAM-Based Compute-in-memory

RRAM devices use a metal–insulator–metal (MIM) stack with a thin metal-oxide layer between two electrodes [7]. They switch by forming or rupturing a conductive filament (CF) in the oxide: a SET voltage expels oxygen ions to create a vacancy chain that creates a low-resistance state (LRS), while a RESET voltage restores ions, breaks the CF, and returns the device to a high-resistance state (HRS). This filamentary switching enables both binary memory and analog resistance tuning for multi-level storage and neuromorphic computing.

Analog/mixed-signal RRAM CIM embeds computation in the memory array (Fig. 1), reducing data movement and energy usage. Digital inputs are converted by DACs into analog voltages applied across memristor rows. Each RRAM cell is a 2-transistor-2-resistor (2T2R) cell, encoding signed neural weights, with the values of -1,0 and 1 being available. Next, column-wise current sums perform parallel vector–matrix multiplications. The results are digitized by ADCs. This architecture sidesteps the Von Neumann bottleneck and reduces the computational complexity to $\mathcal{O}(1)$ compared to $\mathcal{O}(n^2)$ of sequential processing [8].

B. Peripherals

Input/output conversion is handled by ADCs and DACs, each defined by resolution, i.e. the span of analog voltages they cover and the range of corresponding digital codes.

Poster

2378-2250/25/$31.00 ©2025 IEEE
DOI 10.1109/ITC58126.2025.00076

Fig. 2: Test Development Methodology for RRAM Array [9]

TABLE I: Mapping of write background codes to column weight values

Write background Symbol	Weight Description	Applied cell
P	Positive weight	Victim
N	Negative weight	Victim
0	Neutral weight	Victim, Neighbors, Bulk
*	Positive or negative weight	Neighbors, Bulk

Finite resolution introduces quantization error, the difference between the actual and ideal transfer functions. Converters may also suffer stuck-at or missing codes due to differential nonlinearity (DNL), the deviation in step size between adjacent codes, which must stay within 1 Least Significant Bit (LSB) for error-free operation. Summing DNL across all discrete inputs yields integral nonlinearity (INL), a measure of overall deviation from ideal behavior. Finally, from DNL, we generate offset error, when the first two input steps produce the same output (DNL>1 LSB), and gain error, the discrepancy between the ideal and actual output at the highest input level.

III. Proposed Methodology

The process of testing CIM cores follows three steps illustrated in Fig. 2 [9]. The first phase, defect modeling, involves developing defect models in both the memory array and peripheral circuits. In the second step, fault modeling, these models are integrated into the initial netlist and simulated to validate which array & peripheral faults form the theoretical fault space are actually occurring in the circuit. Finally, in the third step, test generation, tests patterns are developed for the validated faults.

IV. Defect Modeling

Defect injection in the 2T2R cell is performed with linear resistive models for both RRAM and CMOS elements. For each possible defect location, resistances from 10Ω to 5 MΩ are swept over ten logarithmic steps via a custom control framework integrated with Cadence Spectre. Redundant defects (e.g., between identical cascode transistors) are removed to speed up simulations. While this approach captures a broad range of resistive faults, it represents only part of the defect spectrum. For instance, defects within the RRAM device are not covered [3].

Peripheral defects are injected into an 8-bit ADC [10], where an NMOS pass transistor converts each cell's bit-line current into a voltage that drives a five-stage voltage-controlled oscillator (VCO), whose pulses feed an asynchronous 8-bit counter. Due to the digital nature of the counter, defect injection is limited to the analog components—the VCO's transistors and the pass transistor—minimizing simulation overhead.

V. Fault Modeling

Simulations are executed on the analog CIM Crossbar architecture of Fig. 1 for 2 columns and 8 rows, under

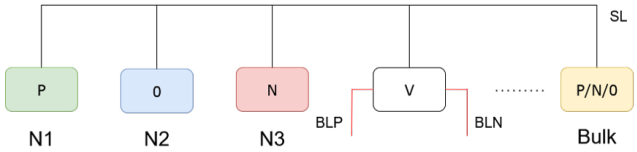

Fig. 3: Input patterns for victim cell and its neighbors on a single column

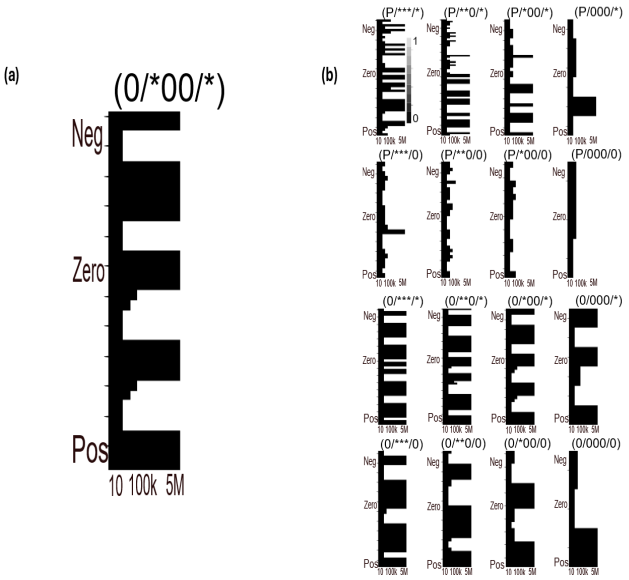

Fig. 4: (a) Fault map example of a SL open 2T2R defect. (b) Zoomed-in example of one of the write background tiles. Dark color indicates unsensitized (0) defect location and white a sensitized (1) one.

the assumption that only a single 2T2R cell (as in Fig. 3e) is defective at any time. Fig. 3 illustrates the applied input patterns per column, where the first 3 neighbors (N1, N2, and N3) are unique RRAM cells each with 3 discrete input-weight product (IN·W) values of [-1, 0, 1]. These 3 neighbors are selected because each 2T2R cell provides three distinct arithmetic states; using fewer would limit the discrete output voltage levels available at the ADC input. The rest of the neighboring cells (Bulk in Fig. 3) have identical IN·W values, which simplifies test generation and reduces simulation overhead. For further simulation easing, only the equivalent IN·W product is considered for all neighboring cells, further reducing their simulation overhead from 9 to 3.

Fault map results are organized by write background, with 24 unique combinations: 3 states for the victim cell (negative, neutral, or positive), 2 for the bulk (positive or neutral only, since only IN·W outputs are simulated and negative products are generated by the BL Drivers), and 4 for all the rest neighboring cells N1-N3 (000 to ***). A (V/N/B) notation is used, where V is the victim cell weights, N are the weights of the 3 distinct neighbors and B is the weight of the bulk cells. Detailed weight assignments are listed in Table I. Fig. 4a shows an example of a specific write background fault map, (0/*00/*). The horizontal axis sweeps through the different defect strengths (10Ω to 5 MΩ), whereas the vertical axis symbolizes the column input patterns, from the most negative

Poster

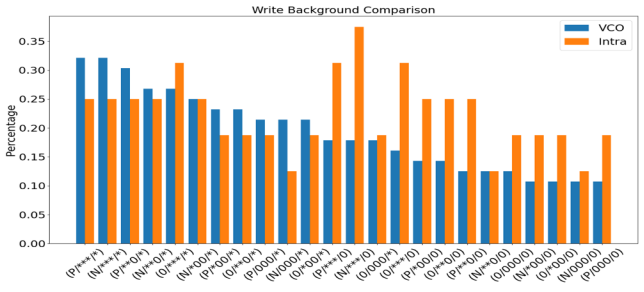

Fig. 5: Normalized Defect coverage per write background

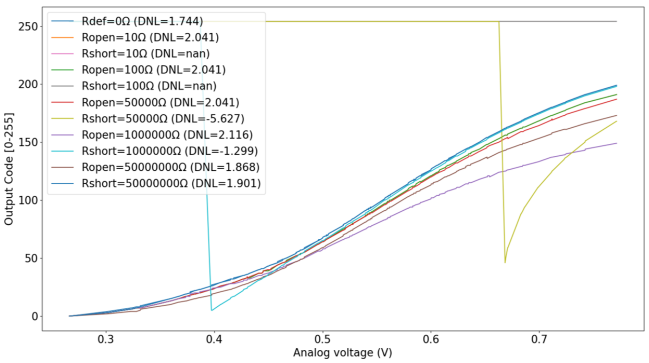

Fig. 6: ADC I/O Characteristics for a short and an open defect case

to the most positive one. Dark color indicates undetected(0) defect location and white a detected(1) one.

Next, we simulate all input patterns for all defect and defect strength combinations, with the results represented as fault maps of Fig. 4b. We group (0/*00/*) and (0/***/0) exhibit low detectability and only produce a few suitable patterns, highlighting the less effective backgrounds for the final test patterns described in Section VI. It should be mentioned that the fault maps of Fig. 4a are already split up per weight group, i.e. write background, thus the vertical axis does not have to encode the IN·W patterns. On the other hand, patterns from backgrounds like (P/***/*), (N/***/*) and (0/000/0) should be favored, since they require only a single RRAM initialization to either all-positive or all-neutral weights, whereas sequences with opposing victim cell and bulk backgrounds demand repeated memory rewrites per victim cell, significantly increasing test time. Furthermore, Fig. 5 shows that backgrounds that require less write operations, such as $(\mathrm{P}/***/*)$ and $(\mathrm{N}/***/*)$, are able to provide full coverage for most oscillator defects, while still offering solid coverage for the 2T2R ones with zero write overhead. Thus, we should include such write backgrounds in the final test setup.

The ADC I/O Transfer functions for 2 different defects are shown in Fig. 6. The short defect sensitizes faults more easily than the open defect, i.e., for a wide range of low analog input values, the exhibited behavior is equivalent to that of a stuck-at fault. As a result, the same defect is responsible for generating both the gain and offset errors. On the other hand, for defects similar to the one of Fig. 6b, a parametric deviation

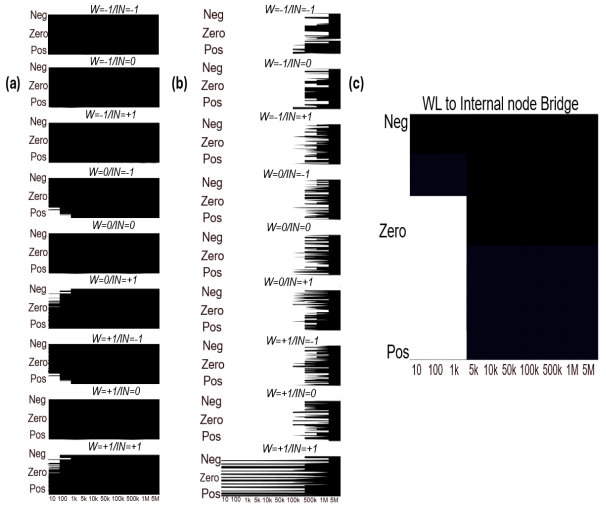

Fig. 7: Fault map comparison (a) Fault map with inference-only operations (b) Fault map with write+read operation combination (c) Bit flip effect due to the extra write operation

is presented that is sensitized mainly by high analog voltage values on the ADC input. We also observe an increasing value of the average DNL as the defect strength increases. DNL values are shown only on the parametric defects of Fig. 6, as non-linearity cannot be defined to an ADC characteristic that has stuck-at, gain and/or offset errors [11]. The depicted open defect, thus, sensitizes the INL and DNL errors.

VI. Test Development

A. Theoretical background

We use the sensitized faults and their fault maps as a part of an Integer Linear Programming (ILP) optimization task for test development. The core objective of this approach is to create a test vector that covers all defects while maintaining the shortest possible sequence of input patterns. At the same time, the final test vector should include the fewest number of write operations to further minimize test time. To solve the ILP problem, we employ Python's SciPy library [12].

For the deployed test sequence, a $(\mathrm{R}_P\mathrm{R}_N{:}\mathrm{BL}_P\mathrm{BL}_N{:}\mathrm{N}_i)$ notation is utilized - with $i \in [1,4]$ - for this work. R_P and R_N are the resistive states of the positive and negative sub-cells of the defective 2T2R cell, BL_P and BL_N are the positive and negative differential inputs provided from the BL driver and N_i are the equivalent IN·W states of the neighboring cells and the bulk.

B. Test for 2T2R Cell Defects

For conclusive structural test development of the 2T2R defects, we first compare of read-only and write+read operations, in Fig. 7 for a RRAM-down to SL bridge. It can be concluded that write operations could lead to bit flips within the 2T2R cell, and thus higher defect coverage. As a result, a direct test comparison between the inference-only and the write-read operations is necessary.

Based on the test procedure of Section VI-A, the inference-only dataset produces a two-length vector of patterns $\{$**(+1-1:+1-1:-1-1-1-1)**,**(+1-1:+1-1:-1+1-1-1)**$\}$ and the write+read

Poster

TABLE II: Test time comparison for the 3 structural tests developed for a 128x128 array.

Test	Patterns	Write/Read	Time(s)	Coverage
Initial	(-1+1:-1+1:-100-1) (+1-1:-1+1:-1-1-1-1) (+1+1:+1+1:+1+1+1+1) (+1-1:+1-1:-1-1-1-1) (+1-1:+1-1:-1+1-1-1)	5N/5N	0.86	93%
Inference+$(\mathrm{P}/***/*)$	(+1-1:-1+1:-1-1-1-1) (+1-1:-1+1:-1+1+1+1) (+1-1:00:+1+1-1-1) (+1-1:+1-1:+1+1+1+1) (+1-1:+1-1:-1-1-1-1) (+1-1:+1-1:-1+1-1-1)	0N/6N	**0.049**	91.3%
Inference+$(\mathrm{N}/***/*)$	(-1+1:-1+1:+1+1+1+-1) (-1+1:00:+1+1+1-1) (-1+1:00:+1+1+1+1) (-1+1:+1-1:-1-1-1-1) (+1-1:+1-1:-1-1-1-1) (+1-1:+1-1:-1+1-1-1)	1N/6N	0.212	**94.6%**

scheme single pattern **{(+1-1:+1-1:0+1+10)}**. However, the latter test leads to an 18% drop in defect coverage (from 76 to 46 detected defect locations). Thus, for unified test purposes, we should keep only the two test patterns generated from the inference-only test strategy, despite the extra test time.

C. Test for ADC

For ADC test defects, we compare the test patterns generated from the entire dataset with the ones from homogeneous write background categories $(\mathrm{P}/***/*)$ and $(\mathrm{N}/***/*)$. That is due to the fact these categories include only positive or negative weights, thus leading to minimal write operations for testing all cells and ADCs of the crossbar array. The highest defect coverage is generated from the initial test set, with the patterns **{(-1+1:-1+1:-100-1),(+1-1:-1+1:-1-1-1-1),(+1+1:+1+1:+1+1+1+1)}**. Next, the patterns from write background $(\mathrm{N}/***/*)$ lead to 1.6% less defect coverage (406 to 414 defect locations) with a test pattern sequence **{(-1+1:-1+1:+1+1+1+-1),(-1+1:00:+1+1+1-1),(-1+1:00:+1+1+1+1),(-1+1:+1-1:-1-1-1-1)}**. The test patterns from $(\mathrm{P}/***/*)$ category **{(+1-1:-1+1:-1-1-1-1),(+1-1:-1+1:-1+1+1+1),(+1-1:00:+1+1-1-1),(+1-1:+1-1:+1+1+1+1)}** lead to a 3.6% defect coverage decrease.

Based on the above results, both $(\mathrm{P}/***/*)$ and $(\mathrm{N}/***/*)$ write background test patterns have a complexity of 0 writes and 4 inference operations per cell and a sub 4% drop-off in defect detection. On the other hand, the nominal test patterns might lead to zero escapes, but that comes with the expense of 5 write operations per considered victim cell, with 3 being on the victim cell itself and two on the neighboring cells. Such a tradeoff between write operations and coverage will be useful for the combined test generation.

D. Combined test

A summary of all the aforementioned combined testing strategies for RRAM and ADC defects is offered in Table II. For calculating the testing time on an assumed 128x128 RRAM crossbar array, for structural test methodologies, we assume a read time of 0.5 μs and write time of 10μs, which were utilized in the deployed simulation setup. The initial generated test on the whole dataset seems to be suboptimal both in terms of test time and defect coverage, due to the increased number of write operations. When combining the inference patterns for RRAM resistive defects with the patterns from the optimal write backgrounds for the oscillator defects,

TABLE III: Test time comparison for structural and functional RRAM tests for a 128x128 array.

Test	Type	Write/Read	Time(s)	Coverage
March-W (1T1R) [13]	Structural	9N/8N	0.86	92%
March C-star [14]	Structural	4N/6N	0.7	91.9%
Analog BIST [6]	Functional	0N/4000N	0.4	94%
RRAMedy [5]	Functional	0N/11000N	0.11	93%
This work	**Structural**	**0N/2N**	**0.016**	**95%**

we notice a significant improvement in test delay. Noticeably, the inference+$(\mathrm{N}/***/*)$ achieves more than 95% coverage with a single write operation, whereas inference+$(\mathrm{P}/***/*)$ test achieves a sub-1ms test time, at the expense of a sub-par coverage compared to all other methods.

VII. Discussion & Conclusion

Table III compares functional and structural test methods for RRAM cell defects. Functional tests use dataset images to detect defects, assuming a $100\,\mu$s delay per image [2]. Our method achieves the highest fault coverage with minimal test time, requiring only a few inference operations. The cited structural tests, by contrast, calculate coverage across the full fault space, extending the work in [15]. While functional tests greatly reduce test time and maintain competitive coverage at the AI Accelerator level, prior approaches rely on high-level fault models (stuck-ats and resistance variation), which are less comprehensive than our circuit-level defect modeling.

This paper presents a column-based testing methodology for analog RRAM CIM MAC circuits, addressing ADC and memory-cell defects. We utilized column-based patterns excel for 2T2R defects, with a hybrid write-inference scheme peripheral test for increased defect coverage. Integrating these techniques at the array level shows initial tests are suboptimal, so combined test sequences trade off coverage versus test time.

References

[1] J. Sun *et al.*, "Analog or digital in-memory computing? benchmarking through quantitative modeling," in *ICCAD*, 2023.

[2] Q. Liu *et al.*, "33.2 a fully integrated analog reram based 78.4tops/w compute-in-memory chip with fully parallel mac computing," in *ISSCC*, 2020, pp. 500–502.

[3] M. Fieback *et al.*, "Defects, fault modeling, and test development framework for rrams," *J. Emerg. Technol. Comput. Syst.*, Apr. 2022.

[4] M. Fieback *et al.*, "Testing scouting logic-based computation-in-memory architectures," in *ETS*, 2020, pp. 1–6.

[5] W. Li *et al.*, "Rramedy: Protecting reram-based neural network from permanent and soft faults during its lifetime," in *ICCD*, 2019.

[6] A. K. Mishra *et al.*, "Built-in functional testing of analog in-memory accelerators for deep neural networks," *Electronics*, 2022.

[7] N. Jain *et al.*, "Resistive random access memory: Materials, filament mechanism, performance parameters and application," in Oct. 2022.

[8] A. Gebregiorgis *et al.*, "An overview of computation-in-memory (cim) architectures," in *DAECS*. Cham, 2024.

[9] C. Wang *et al.*, "Defects, fault modeling, and test development framework for fefets," in *ITC*, 2024.

[10] M. Mayahinia *et al.*, "A voltage-controlled, oscillation-based adc design for computation-in-memory architectures using emerging rerams," *J. Emerg. Technol. Comput. Syst.*, Mar. 2022.

[11] F. Correa Alegria *et al.*, "Standard histogram test precision of adc gain and offset error estimation," *IEEE TIM*, 2007.

[12] P. Virtanen *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, 2020.

[13] Y. Luo *et al.*, "A high fault coverage march test for 1t1r memristor array," in *EDSSC*, IEEE, 2017.

[14] C.-Y. Chen *et al.*, "Rram defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *IEEE TC*, 2014.

[15] H. Xun *et al.*, "Data background-based test development for all interconnect and contact defects in rrams," in *ETS*, 2023, pp. 1–6.

Poster